Towards a Logic for Performance and Mobility FULL VERSION

Rocco De Nicola[1], Joost-Pieter Katoen[2], Diego Latella[3], and Mieke Massink[3]

[1] Univ. of Firenze, Firenze, Italy
denicola@dsi.unifi.it
[2] Univ. of Twente, Enschede, The Netherlands
and RWTH Aachen, Aachen, Germany
katoen@cs.utwente.nl
[3] CNR, Istituto di Scienza e Tecnologie dell'Informazione, Pisa, Italy
{Diego.Latella,Mieke.Massink}@isti.cnr.it

Abstract. KLAIM is an experimental language designed for modeling and programming distributed systems composed of mobile components where distribution awareness and dynamic system architecture configuration are key issues. STOCKLAIM [13] is a Markovian extension of the core subset of KLAIM which includes process distribution, process mobility, asynchronous communication, and site creation. By enriching process actions with specific rates characterizing the exponential distributions modelling the durations of such actions, the extension makes it possible to integrate the modelling of quantitative aspects of mobile systems—e.g. performance—with the functional specification of such systems. In this paper, $MoSL$, a temporal logic for STOCKLAIM is proposed which addresses and integrates both the issues of distribution awareness and mobility and those concerning stochastic behaviour of systems. The satisfiability relation is formally defined over (states-/transition-)labelled Markov Chains. A translation of the logic to action-based CSL is provided which allows to use of existing aCSL model-checkers for the verification of STOcKLAIM models against $MoSL$ properties. Examples of applications are provided as well.

1 Introduction

During the last decades, computer systems have changed from isolated static devices to machines that are highly interconnected to perform tasks in a cooperative and coordinated manner. These modern, complex distributed systems—also known as *global or network-aware computers* [8]—are highly dynamic and have to deal with frequent changes of the network environment. New features such as distribution awareness and code mobility play a prominent role in the concept of global computing. Dedicated programming and specification formalisms have been developed that can deal with issues such as (code and agent) mobility, remote execution, and privacy and security aspects (e.g., data integrity). Prominent examples of such languages and frameworks are, among others, Obliq [7], Seal [10], and KLAIM [12].

Due to their enormous size— networks typically consist of thousands or even millions of nodes— and their strong reliance on mobility and interaction, performance and dependability issues of global computers are of utmost importance for "network-aware computing". Spontaneous computer crashes may easily lead to failure of remote execution or process movement, while spurious network hick ups may cause loss of code fragments or unpredictable delays. The presence of such random phenomena implies that correctness of practical global programs and their privacy guaranties are no longer rigid notions ("either it is safe or it is not"), but have a less absolute nature ("in 97% of the cases, privacy can be ensured"). The intrinsic complexity of global computers, though, complicates the assessment of these issues severely. Systematic methods, techniques and tools are therefore needed to establish performance and dependability requirements and guarantees.

This paper reports on our initial attempts towards such systematic methods. We consider an extension of the core subset of KLAIM [5] with random delays, as proposed in [13]. This yields an

integrated specification language supporting major global computing paradigms such as process mobility, process distribution, asynchronous communication of node names and processes through shared local repositories (i.e., tuple spaces), and dynamic node creation, as well as randomly delayed activities. For the sake of simplicity we restrict to exponential delays. This facilitates the use of existing numerical techniques and software tools. The generalization of our approach toward general distributions can be performed along the lines of [11].

The main contribution of this paper is a state/action-based, branching-time temporal logic that permits specification of properties of stochastic core KLAIM terms. The logic permits to consider the mobile and distributed aspects of global computing, its performance and dependability requirements, and their combination. It allows, for instance, to state that "in equilibrium, the probability that a piece of code currently at site ℓ eventually ends up at site ℓ' exceeds 0.9" and "a certain network configuration can be reached without ever dynamically creating a new site". The syntax and semantics of the logic are detailed, and a mapping is provided for a large fragment of it onto action-based CSL (aCSL [23]). The relation between aCSL and CSL (Continuous Stochastic Logic [1, 2]) is similar to that between CTL and action-based CTL [15]. Our mapping facilitates the use of the existing model checker ETMCC [24] for the automated verification of STOCKLAIM programs. ETMCC provides efficient algorithms for model-checking Continuous Time Markov Chains and it has been used for the validation of a number of real-life applications. We illustrate our approach by modelling the spreading of a virus through a network and verifying stochastic properties such as "the probability that the virus is spread to location ℓ within t time-units is less than 10^{-4}". Several (temporal) logics have been proposed which aim at describing properties of systems related either to mobility ([4, 14, 6, 9, 18, 26] among others) or to probabilistic/stochastic behaviour (e.g. [20, 21, 1, 2, 23]). To the best of our knowledge, the present paper is the first approach towards a probabilistic logic for mobility.

The paper is further organized as follows. Section 2 presents the STOCKLAIM language and its semantics in terms of action-labelled continuous-time Markov chains (CTMCs). Section 3 introduces the "performobility" logic, its syntax and semantics, and examples that illustrate its expressiveness. Section 4 discusses the issues of translating the logic into aCSL and its impact on the model-checking algorithms. Section 5 presents the virus spreading case study and its verification results. Section 6 concludes the paper. The detailed proofs can be found in Appendix B.

2 StocKlaim

In the following we shall briefly recall the STOCKLAIM language definition. In [13] the definition of the language is dealt with in full detail and a thorough discussion of the motivations for all design choices is presented. We refer the interested reader to the above mentioned papers.

2.1 Syntax of StocKlaim

Let $\mathcal{L}$, ranged over by $l, l', l_1, \ldots$, be a set of *localities*, $\mathcal{U}$, ranged over by $u, u', u_1, \ldots$, a set of *locality variables*, $\mathcal{A}$, ranged over by $A, A', A_1, \ldots$ a set of *process variables*, and $\mathcal{R}$, ranged over by $r, r', r_1, \ldots$ a set of *rate names*. We assume that the above sets are mutually disjoint. Moreover, we let ℓ, ℓ' range over $\mathcal{L} \cup \mathcal{U}$.

The syntax of STOCKLAIM, is given in Table 1. A *network* node $l :: \langle l' \rangle$ intuitively models that value $\langle l' \rangle$ is stored, or located, in node, or locality, l. Similarly, for process P, $l :: \langle P \rangle$ means that P is stored in l as a piece of data. On the other hand, $l :: P$ indicates that process P is running at locality l. Complex networks are built from simpler ones by means of the *network parallel operator* $\|$. Given network N, the set of values located at locality l consists of those l' and P such that $l :: \langle l' \rangle$ or $l :: \langle P \rangle$ occurs in N. The set of processes running at locality l coincides with all those P such that $l :: P$ occurs in N. The intuition behind action prefix $(a, r).P$ is that the execution time of action a is distributed exponentially with rate specified by *rate name* r. Rate names are mapped to rate values by means of binding functions, which are (partial) functions from $\mathcal{R}$ to $\mathbb{R}^+$. Actions can be used for uploading/downloading of data as well as processes to/from specific

$N ::=$ NETWORKS	$P ::=$ PROCESSES	$a ::=$ ACTIONS	$T ::=$ TEMPLATES
$l :: \langle l' \rangle$	**nil**	**out** $\ell' @ \ell$	l
$\mid \quad l :: \langle P \rangle$	$\mid \quad (a, r).P$	$\mid$ **out** $P @ \ell$	$\mid \quad !u$
$\mid \quad l :: P$	$\mid \quad P + P$	$\mid$ **in** $T @ \ell$	$\mid \quad !A$
$\mid \quad N \parallel N$	$\mid \quad P \mid P$	$\mid$ **eval** $P @ \ell$	
	$\mid \quad A$	$\mid$ **newloc** u	

Table 1. Syntax of StocKlaim

localities. Process (remote) evaluation is also possible as well as the creation of new localities. Compound processes are built using the usual operators like choice, parallel and instantiation.

2.2 Semantics of StocKlaim

The operational semantics definition characterizes a reduction relation over (network) configurations. A configuration is a triple (L, β, N)—henceforth written as $L, \beta \vdash N$—where L is a finite set of localities, $\beta : \mathcal{R} \mapsto \mathbb{R}^+$ is a mapping from rate names to rates, with $(dom\ \beta)$—the domain of β—also finite, and N a StocKlaim network expression. We use L_c (resp. β_c and N_c) for L (resp. β and N) in configuration $c = L, \beta \vdash N$. Let $(\mathsf{Loc}\ N)$ denote the set of all localities occurring free[1] in N and $(\mathsf{Rat}\ N)$ be the set of all rate names occurring in N. A *well formed* network specification is a network configuration $L, \beta \vdash N$ where:

- $(\mathsf{Loc}\ N) \subseteq L$, and $(\mathsf{Rat}\ N) \subseteq (dom\ \beta)$;
- each process variable A occurring free in N has a *single defining equation* of the form $A \stackrel{\Delta}{=} P$, where P may contain occurrences of A and other process names; occurrences of A on the right part are always guarded, i.e. prefixed by a stochastic action;
- all rate names occurring in N are *distinct* and for expressions of the form **in** $!A @ l'.P$, (i) there exists *at most one* free occurrence of A in P which is not the first argument of an **out** or **eval** operator, and (ii) there exists no defining equation for A.

In the sequel we shall consider only well formed specifications and we shall use the shorthand (N, β) for $(\mathsf{Loc}\ N), \beta \vdash N$. Rate name uniqueness requirements ensure that whenever there is more than one way to perform the same action, the *total* rate for such action—i.e. that obtained considering the contribution of all the different ways to perform the action—will be taken into account, as discussed in [13]. The reduction relation $\longrightarrow$ is the smallest relation induced by the rules of Table 2. Notice that the reduction relation of Table 2 is slightly different from that presented in [13] since each transition is labeled not only with the associated rate name, but also with the actual action which has taken place and the locality where it has been executed—in [13] the label consists of the rate name alone. The reason for such an enrichment of transition labels is that the logic which we define in the present paper is (also) *action-based*. We let ACT denote the set of all actions which can be generated by StocKlaim processes. In order to apply the rules of Table 2 it is often necessary to rearrange configurations according to the Structural Congruence, which is the smallest relation satisfying the laws given in Table 8 in the appendix. Notice that the rules and laws are designed in such a way that rate name uniqueness is preserved by their application, by means of function RN defined in Fig. 6, also in the appendix.

The operational semantics of a network specification (N, β) are defined, like in [13], as the LTS $TS(N, \beta) \stackrel{\text{def}}{=} (C, \Lambda, \rightarrow, c_0)$ where C is the set of (the representatives under $\equiv$ of) the configurations reachable from (the representative under $\equiv$ of) the initial state $c_0 \stackrel{\text{def}}{=} (\mathsf{Loc}\ N), \beta \vdash N$, via the transition relation $\rightarrow \subseteq C \times \Lambda \times C$. The *label set* Λ is a subset of $\mathcal{L} \times ACT \times \mathcal{R}$.

[1] The definition of free and bound variables is the same as cKlaim, as it can be found e.g. in [19].

$$L, \beta \vdash l :: (\mathbf{out}\ l'' \ @\ l', r).P \parallel l' :: P' \xrightarrow{(l, \mathbf{out}\ l''\ @\ l', r)} L, \beta \vdash l :: P \parallel l' :: P' \parallel l' :: \langle l'' \rangle \quad \text{(OUTL)}$$

$$L, \beta \vdash l :: (\mathbf{out}\ Q \ @\ l', r).P \parallel l' :: P' \xrightarrow{(l, \mathbf{out}\ Q'\ @\ l', r)} L, \beta' \vdash l :: P \parallel l' :: P' \parallel l' :: \langle Q' \rangle \quad \text{(OUTP)}$$
where $(Q', \beta') = \mathsf{RN}(Q, \beta)$

$$L, \beta \vdash l :: (\mathbf{in}\ T \ @\ l', r).P \parallel l' :: \langle l'' \rangle \xrightarrow{(l, \mathbf{in}\ l''\ @\ l', r)} L, \beta \vdash l :: P\Theta \parallel l' :: \mathbf{nil} \quad \text{(INL)}$$
where $\Theta = \begin{cases} [l''/u], & \mathbf{if}\ \ T = !u \\ \epsilon, & \mathbf{if}\ \ T = l'' \end{cases}$

$$L, \beta \vdash l :: (\mathbf{in}\ (!A) \ @\ l', r).P \parallel l' :: \langle P' \rangle \xrightarrow{(l, \mathbf{in}\ P'\ @\ l', r)} L, \beta \vdash l :: P[P'/A] \parallel l' :: \mathbf{nil} \quad \text{(INP)}$$

$$L, \beta \vdash l :: (\mathbf{newloc}\ u, r).P \xrightarrow{(l, \mathbf{newloc}\ l', r)} L \cup \{l'\}, \beta \vdash\ l :: P[l'/u] \ \parallel\ l' :: \mathbf{nil} \quad \text{(NLC)}$$
where $l' = \mathbf{choose}\ l_1 \in \mathcal{L} \setminus L$

$$L, \beta \vdash l :: (\mathbf{eval}\ Q \ @\ l', r).P \parallel l' :: P' \xrightarrow{(l, \mathbf{eval}\ Q'\ @\ l', r)} L, \beta' \vdash l :: P \parallel l' :: P' \mid Q' \quad \text{(EVA)}$$
where $(Q', \beta') = \mathsf{RN}(Q, \beta)$

$$\frac{L, \beta' \vdash l :: P' \xrightarrow{(l,a,r)} L', \beta'' \vdash N}{L, \beta \vdash l :: A \xrightarrow{(l,a,r)} L', \beta'' \vdash N} \quad \text{(PIN)}$$
where $A \stackrel{\Delta}{=} P$ and $(P', \beta') = \mathsf{RN}(P, \beta)$

$$\frac{L, \beta \vdash l :: P_1 \parallel N \xrightarrow{(l,a,r)} L', \beta' \vdash l :: P' \parallel N'}{L, \beta \vdash l :: P_1 + P_2 \parallel N \xrightarrow{(l,a,r)} L', \beta' \vdash l :: P' \parallel N'} \quad \text{(CHO)}$$

$$\frac{L, \beta \vdash N_1 \xrightarrow{(l,a,r)} L', \beta' \vdash N'}{L, \beta \vdash N_1 \parallel N_2 \xrightarrow{(l,a,r)} L', \beta' \vdash N' \parallel N_2} \quad \text{(PAR)}$$

$$\frac{\begin{array}{c} L, \beta \vdash N \equiv L_1, \beta_1 \vdash N_1, \beta_1 = \theta^{-1}\beta \\ L_1, \beta_1 \vdash N_1 \xrightarrow{(l,a,\theta r)} L_2, \beta_2 \vdash N_2, \\ L_2, \beta_2 \vdash N_2 \equiv L', \beta' \vdash N' \end{array}}{L, \beta \vdash N \xrightarrow{(l,a,r)} L', \beta' \vdash N'} \quad \text{(STC)}$$

Table 2. Reduction Rules of StocKlaim

The derivatives as well as the transition relation $\rightarrow$ are defined in the usual way using the rules and laws given in the tables mentioned above. We say that such an LTS is finite if and only if it is finitely branching and set C is finite. Whenever such LTS is finite a finite CTMC can easily be derived from it. There are several ways for assuring finiteness of the LTSs that are automatically generated from higher level specifications, like process algebras. Some rely on syntactical restrictions, like avoiding certain (combinations of) operators, and they have been studied extensively in the context of traditional process algebra (see e.g. [17]). Others, typically used in the context of verification tools design, are based on the introduction of constraints on certain kinds of resources, e.g. buffer sizes and data value domains, in the definition of the operational semantics. The latter approach seems to be most suitable for StocKlaim. For instance a limit can be imposed on the maximum number of values which can be stored in a single node. We leave the details of these issues for further study.

2.3 From StocKlaim to CTMCs

CTMCs have been extensively studied in the literature (a comprehensive treatment can be found in [25]; we suggest [22] for a gentle introduction). For the purposes of the present paper we will use the notion of *action-labelled CTMC*, as defined for instance in [23]:

Definition 1. *An* Action-labelled *CTMC (AMC), $\mathcal{M}$, is a triple $(S, Act, \mapsto)$ where S is a set of states, Act is a set of actions, and $\mapsto \subseteq S \times (Act \times \mathbb{R}^+) \times S$ is the transition relation.*

In the sequel we consider only finite AMCs, i.e. finitely branching AMCs with a finite number of states. Moreover, *Act*, ranged over by $\gamma, \gamma', \gamma_1$ is a set of *located* actions (l, a) where a is a StocKlaim action and l the locality where a takes place (we will omit the word "located" for simplicity, when this will not cause confusion). Transition $s \stackrel{\gamma,\lambda}{\mapsto} s'$ means that the process can move from state s to state s' while performing action γ with an execution time determined by an exponential distribution with rate λ. We say that $s \in S$ is *absorbing* if and only if there is no s', γ and λ such that $s \stackrel{\gamma,\lambda}{\mapsto} s'$. Notice that according to the above definition AMCs can have self-loops. The presence of such transitions do not affect standard Markov Chain measures and allows for a natural definition of the action-based temporal operators in the logic.
The AMC associated to a StocKlaim network specification is defined below:

Definition 2. *Given a network specification (N, β) and assuming $TS(N, \beta) = (C, \Lambda, \rightarrow, c_0)$ finite, the AMC $(S, Act, \mapsto)$ associated to the specification and denoted by $AMC(N, \beta)$, is such that $S = C$, $Act \subseteq (\mathcal{L} \times ACT) \times \mathbb{R}^+$ and for all $c, c' \in S$, $c \stackrel{(l,a),\lambda}{\mapsto} c'$ if and only if $0 \neq \lambda = \sum_{c \xrightarrow{l,a,r} c'} (\beta_c\ r)$.*

We close this section with a few definitions which we will use in the next section. Given AMC $(S, Act, \mapsto)$, $\Gamma \subseteq Act$, and $s, s' \in S$, let $\mathbf{R}_\Gamma(s, s')$, $\mathbf{E}(s)$, and $\mathbf{P}_\Gamma(s, s')$ be defined as follows:

$$\mathbf{R}_\Gamma(s, s') \stackrel{\text{def}}{=} \sum_{\gamma \in \Gamma} \{\lambda \mid s \stackrel{\gamma,\lambda}{\mapsto} s'\}$$
$$\mathbf{E}(s) \stackrel{\text{def}}{=} \sum_{s' \in S} \mathbf{R}_{Act}(s, s')$$
$$\mathbf{P}_\Gamma(s, s') \stackrel{\text{def}}{=} \mathbf{R}_\Gamma(s, s')/\mathbf{E}(s)$$

That is, $\mathbf{R}_\Gamma(s, s')$ is the cumulative rate of moving from state s to state s' via any action in Γ, $\mathbf{E}(s)$ is the rate of the total sojourn time in s and $\mathbf{P}_\Gamma(s, s')$ the probability of moving from s to s' performing an action in set Γ. The definition of *Paths* over an AMC follows:

Definition 3 (Paths). *Given AMC $\mathcal{M} = (S, Act, \mapsto)$,*

i) an infinite path *σ over $\mathcal{M}$ is a sequence $s_0(\gamma_0, t_0)s_1(\gamma_1, t_1)s_2(\gamma_2, t_2)\ldots$ such that, for all $i \geqslant 0$, $s_i \in S, \gamma_i \in Act$, $t_i \in \mathbb{R}^+$ and $\mathbf{R}_{\{\gamma_i\}}(s_i, s_{i+1}) > 0$. For infinite path σ and $i \geqslant 0$ we let $\mathsf{len}(\sigma) = \infty$, $\sigma[i] = s_i$, $\alpha(\sigma, i) = \gamma_i$, and $\delta(\sigma, i) = t_i$; for $t \in \mathbb{R}_{\geqslant 0}$ and i the smallest index such that $t \leqslant \sum_{j=o}^{i} t_j$ we let $\sigma(t) = \sigma[i]$.*

ii) A finite path *σ over $\mathcal{M}$ is a sequence $s_0(\gamma_0, t_0)s_1(\gamma_1, t_1)s_2(\gamma_2, t_2)\ldots s_{l-1}(\gamma_{l-1}, t_{l-1})s_l$ such that s_l is absorbing and, for $0 \leqslant i < l$, $s_i \in S, \gamma_i \in Act$, $t_i \in \mathbb{R}^+$ and $\mathbf{R}_{\{\gamma_i\}}(s_i, s_{i+1}) > 0$. For finite σ, $\mathsf{len}(\sigma) = l$, $\sigma[i]$, $\alpha(\sigma, i)$, and $\delta(\sigma, i)$ are defined only for $i \leqslant l$; they are defined as above for $i < l$, $\sigma[l] = s_l$, $\alpha(\sigma, l) = \gamma_l$, and $\delta(\sigma, l) = \infty$; furthermore, for $t > \sum_{j=o}^{l-1} t_j$ we let $\sigma(t) = \sigma[l]$, otherwise $\sigma(t)$ is defined as for the case σ infinite.*

For any state s of an AMC $\mathcal{M}$, we let $\mathsf{Paths}(s)$ denote the set of all paths $s_0(\gamma_0, t_0)s_1(\gamma_1, t_1)s_2(\gamma_2, t_2)\ldots$ over $\mathcal{M}$ with $s_0 = s$. A Borel space can be defined over $\mathsf{Paths}(s)$, together with its associated probability measure Pr, which is a slight extension of that defined e.g in [2] in order to take both states and actions into consideration:

Definition 4 (Borel Space). *Given AMC $\mathcal{M} = (S, Act, \mapsto)$, let $s_0, \ldots, s_k \in S$, $\gamma_0, \ldots, \gamma_{k-1} \in Act$, with $\mathbf{R}_{\{\gamma_i\}}(s_i, s_{i+1}) > 0$ for $0 \leqslant i < k$, and $I_0, \ldots, I_{k-1}$ non-empty intervals in $\mathbb{R}_{\geqslant 0}$. Then $C(s_0, \gamma_0, I_0, \ldots, s_{k-1}, \gamma_{k-1}, I_{k-1}, s_k)$ is the* cylindric set *consisting of all paths $\sigma \in \mathsf{Paths}(s_0)$ with $\sigma[i] = s_i$ for $0 \leqslant i \leqslant k$ and, for $0 \leqslant i < k$, $\alpha(\sigma, i) = \gamma_i$ and $\delta(\sigma, i) \in I_i$. For any $s \in S$, let*

$\mathcal{F}(\mathsf{Paths}(s))$ be the smallest σ-algebra which contains all sets $C(s_0, \gamma_0, I_0, \ldots, s_{k-1}, \gamma_{k-1}, I_{k-1}, s_k)$ for $s_0, \ldots, s_k \in S$, $s_0 = s$, and $\gamma_0, \ldots, \gamma_{k-1}$ with $\mathbf{R}_{\{\gamma_i\}}(s_i, s_{i+1}) > 0$ for $0 \leqslant i < k$, and $I_0, \ldots, I_{k-1}$ non-empty intervals in $\mathbb{R}_{\geqslant 0}$. The probability measure $\Pr$ on $\mathcal{F}(\mathsf{Paths}(s))$ is the unique measure defined by induction on k as follows:

$$\begin{array}{l} \Pr(C(s_0)) \stackrel{def}{=} 1 \\ \Pr(C(s_0, \gamma_0, I_0, \ldots, s_{k-1}, \gamma_{k-1}, I_{k-1}, s_k)) \stackrel{def}{=} \\ \quad \Pr(C(s_0, \gamma_0, I_0, \ldots, s_{k-1})) \cdot \mathbf{P}_{\{\gamma_{k-1}\}}(s_{k-1}, s_k) \cdot \left(e^{-\mathbf{E}(s_{k-1}) \cdot \underline{I}} - e^{-\mathbf{E}(s_{k-1}) \cdot \overline{I}} \right) \text{ for } k > 0 \end{array}$$

where $\underline{I} = \mathsf{inf}\, I_{k-1}$ and $\overline{I} = \mathsf{sup}\, I_{k-1}$ (we let $e^{-\mathbf{E}(s_{k-1}) \cdot \overline{I}} \stackrel{def}{=} 0$ if $\mathbf{E}(s_{k-1}) > 0$ and $\overline{I} = \infty$).

3 *MoSL*: a logic for StocKlaim

In this section, we present *MoSL*, a logic for the integrated specification of *functional*—i.e. qualitative—as well as *quantitative*—e.g. performance—properties of *mobile* systems modeled using StocKlaim.

3.1 Syntax of *MoSL*

Let $\mathcal{L}$, $\mathcal{U}$, and $\mathcal{A}$ be defined as in Sect. 2.1, $p \in [0,1]$ a probability value, $\bowtie \in \{<, \leqslant, \geqslant, >\}$, and $t \in \mathbb{R}^+ \cup \{\infty\}$. The syntax of *MoSL* is defined Table 3.

$\Phi ::=$ State Formulae	$\aleph ::=$ Atomic Propositions	$\xi ::=$ Action Formulae
tt	$A@\ell$	tt
$\aleph$	$\langle \ell \rangle @ \ell$	$\neg\, \xi$
$\neg\, \Phi$	$\langle A \rangle @ \ell$	$\xi \vee \xi$
$\Phi \vee \Phi$		$\ell : \mathbf{out}\ \ell\ @\ \ell$
$\mathcal{S}_{\bowtie p}(\Phi)$	$\varphi ::=$ Path Formulae	$\ell : \mathbf{out}\ A\ @\ \ell$
$\mathcal{P}_{\bowtie p}(\varphi)$	$\Phi\, {}_{\xi}\mathbf{U}^{<t}_{\xi}\, \Phi$	$\ell : \mathbf{in}\ \ell\ @\ \ell$
		$\ell : \mathbf{in}\ A\ @\ \ell$
	$\Phi\, {}_{\xi}\mathbf{U}^{<t}\, \Phi$	$\ell : \mathbf{eval}\ A\ @\ \ell$
		$\ell : \mathbf{newloc}\ u$

Table 3. Syntax of *MoSL*

MoSL essentially extends aCSL in two dimensions. First of all, besides the trivial proposition tt, *MoSL* state formulae include those built from $A@l$ (resp. $\langle l' \rangle @ l$, $\langle A \rangle @ l$) modeling the fact that process A is executing at locality l (resp. value l' or process A is stored at locality l). Locality variables can be used as well and they will be bound by means of action formulae of the form $\ell : \mathbf{newloc}\ u$, as we will see later.

Moreover, requirements on actions are not expressed by sets of labels, as in aCSL, but they are (boolean combinations of) action propositions directly reflecting the actions of StocKlaim.

We recall that $\mathcal{P}_{\bowtie p}(\varphi)$ asserts that the probability measure of the set of paths stisfying φ meets the bound $\bowtie p$ while $\mathcal{S}_{\bowtie p}(\Phi)$ means that the steady-state probability for the set of states satisfying Φ meets the bound $\bowtie p$. The path formula $\Phi\, {}_{\xi}\mathbf{U}^{<t}\, \Phi'$ is fulfilled by a path if a Φ'-state is eventually reached by passing only through Φ-states before, while taking only transitions the actions of which satisfy ξ; besides, going from the beginning of the path till reaching the Φ'-state should last at most t time units. The formula $\Phi\, {}_{\xi}\mathbf{U}^{<t}_{\xi'}\, \Phi'$ requires moreover that (i) a move to a Φ'-state is actually made and that (ii) this transition is labeled by an action satisfying ξ'.

As it is clear from the syntactical definition of $MoSL$, formulae may contain both locality variables and process variables. While process variables will be used as non-interpreted symbols in the definition of $MoSL$ semantics, an action formula like $l : \mathbf{newloc}\ u$ may implicitly bind locality variable u, when interpreted over an action of a path. Consequently, function Fr, defined in Fig.1, characterizes the set of free (locality) variables of $MoSL$ formulae.

$$
\begin{array}{ll}
\mathsf{Fr}(\mathtt{tt}) \stackrel{\text{def}}{=} \varnothing & \mathsf{Fr}(\ell : \mathbf{out}\ \ell' @ \ell'') \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) \cup \mathsf{Fr}(\ell') \cup \mathsf{Fr}(\ell'') \\
\mathsf{Fr}(A@\ell) \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) & \mathsf{Fr}(\ell : \mathbf{out}\ A @ \ell') \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) \cup \mathsf{Fr}(\ell') \\
\mathsf{Fr}(\langle \ell' \rangle @\ell) \stackrel{\text{def}}{=} \mathsf{Fr}(\ell') \cup \mathsf{Fr}(\ell) & \mathsf{Fr}(\ell : \mathbf{in}\ \ell' @ \ell'') \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) \cup \mathsf{Fr}(\ell') \cup \mathsf{Fr}(\ell'') \\
\mathsf{Fr}(\langle A \rangle @\ell) \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) & \mathsf{Fr}(\ell : \mathbf{in}\ A @ \ell') \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) \cup \mathsf{Fr}(\ell') \\
\mathsf{Fr}(\neg \Phi) \stackrel{\text{def}}{=} \mathsf{Fr}(\Phi) & \mathsf{Fr}(\ell : \mathbf{eval}\ A @ \ell') \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) \cup \mathsf{Fr}(\ell') \\
\mathsf{Fr}(\Phi \vee \Psi) \stackrel{\text{def}}{=} \mathsf{Fr}(\Phi) \cup \mathsf{Fr}(\Psi) & \mathsf{Fr}(\ell : \mathbf{newloc}\ u) \stackrel{\text{def}}{=} \mathsf{Fr}(\ell) \\
\mathsf{Fr}(\mathcal{S}_{\bowtie p}(\Phi)) \stackrel{\text{def}}{=} \mathsf{Fr}(\Phi) & \mathsf{Fr}(\Phi\ {}_{\xi}\mathbf{U}^{<t}_{\eta}\ \Psi) \stackrel{\text{def}}{=} \mathsf{Fr}(\Phi) \cup \mathsf{Fr}(\xi) \cup \mathsf{Fr}(\eta) \cup \\
\mathsf{Fr}(\mathcal{P}_{\bowtie p}(\varphi)) \stackrel{\text{def}}{=} \mathsf{Fr}(\varphi) & \qquad \mathsf{Fr}(\Psi) \setminus (\mathsf{Bnd}(\xi) \cup \mathsf{Bnd}(\eta)) \\
\mathsf{Fr}(\xi \vee \eta) \stackrel{\text{def}}{=} \mathsf{Fr}(\xi) \cup \mathsf{Fr}(\eta) & \mathsf{Fr}(\Phi\ {}_{\xi}\mathbf{U}^{<t}\ \Psi) \stackrel{\text{def}}{=} \mathsf{Fr}(\Phi) \cup \mathsf{Fr}(\xi) \cup (\mathsf{Fr}(\Psi) \setminus (\mathsf{Bnd}(\xi)))
\end{array}
$$

where we let $\mathsf{Fr}(l) \stackrel{\text{def}}{=} \varnothing$ for all localities $l \in \mathcal{L}$, $\mathsf{Fr}(u) \stackrel{\text{def}}{=} \{u\}$ for all locality *variables* $u \in \mathcal{U}$. Function Bnd is defined below and characterize the set of locality variables which are *bound* by an action formula:

$$
\begin{array}{ll}
\mathsf{Bnd}(\mathtt{tt}) \stackrel{\text{def}}{=} \varnothing & \mathsf{Bnd}(\ell : \mathbf{in}\ \ell' @ \ell'') \stackrel{\text{def}}{=} \varnothing \\
\mathsf{Bnd}(\neg \xi) \stackrel{\text{def}}{=} \varnothing & \mathsf{Bnd}(\ell : \mathbf{in}\ A @ \ell') \stackrel{\text{def}}{=} \varnothing \\
\mathsf{Bnd}(\xi \vee \eta) \stackrel{\text{def}}{=} \mathsf{Bnd}(\xi) \cap \mathsf{Bnd}(\eta) & \mathsf{Bnd}(\ell : \mathbf{eval}\ A @ \ell') \stackrel{\text{def}}{=} \varnothing \\
\mathsf{Bnd}(\ell : \mathbf{out}\ \ell' @ \ell'') \stackrel{\text{def}}{=} \varnothing & \mathsf{Bnd}(\ell : \mathbf{newloc}\ u) \stackrel{\text{def}}{=} \{u\} \\
\mathsf{Bnd}(\ell : \mathbf{out}\ A @ \ell') \stackrel{\text{def}}{=} \varnothing &
\end{array}
$$

Fig. 1. Free Locality Variables in $MoSL$

In the following only *well-formed* formulae will be considered. We say that a formula Φ is if and only if it contains no free variables (i.e. $\mathsf{Fr}(\Phi) = \varnothing$) and each action formula ξ occurring in Φ binds at most one locality variable.

A few comments on the definition of functions Fr and Bnd are on demand. First, notice that free occurrences of u in Φ remain free also in $\Phi\ {}_{\xi}\mathbf{U}^{<t}_{\eta}\ \Psi$ (and $\Phi\ {}_{\xi}\mathbf{U}^{<t}\ \Psi$), *even* if they are bound by ξ (i.e. they are elements of $\mathsf{Bnd}(\xi)$). As we will see in the definition of the semantics of $MoSL$, a path formula of the logic will be interpreted, via relation $\models$, over a path σ as defined in Sec. 2. Now, for defining $\sigma \models \Phi\ {}_{\xi}\mathbf{U}^{<t}\ \Psi$ it is necessary to know if $\sigma[0] \models \Phi$; on the other hand, any binding of u by means of a sub-formula of ξ of the form $\ell : \mathbf{newloc}\ u$ will take effect *only* from $\sigma[1]$ on, i.e. after the "first transition" has taken place. Similarly, we get $u \in \mathsf{Fr}(\Phi\ {}_{\xi}\mathbf{U}^{<t}_{\eta}\ \Psi)$ if $u \in \mathsf{Fr}(\eta)$, even in the case $u \in \mathsf{Fr}(\xi)$. Another issue which deserves some attention is the definition of $\mathsf{Bnd}(\xi \vee \eta)$. Consider the formula $\mathtt{tt}\ {}_{\mathtt{tt}}\mathbf{U}^{<\infty}_{\xi}\ \Phi$, where ξ is the formula $l : \mathbf{newloc}\ u \vee l : \mathbf{out}\ l @ l$ which informally means that a state where Φ holds will eventually be reached via a transition corresponding to the execution of action $\mathbf{newloc}\ u$ or $\mathbf{out}\ l @ l$ (in locality l); in the latter case, no binding occurs for u and this would make it wrong to use the variable in Φ. Consequently, in such cases Bnd returns the empty set. Conversely, u can safely occur in Φ in the above formula if ξ stands for $l_1 : \mathbf{newloc}\ u \vee l_2 : \mathbf{newloc}\ u$. Finally, notice that $\mathsf{Bnd}(\neg \xi)$ is the empty set since variables u occurring in formulae like $l : \mathbf{newloc}\ u$ in ξ are actually *not* bound if $\neg \xi$ is satisfied. Strictly speaking, if for instance $\neg\neg l : \mathbf{newloc}\ u$ is satisfied, then u should be bound to the locality l' appearing in the transition. A proper study of the contexts where the binding should take place and those in which this is not the case could allow for a more refined definition of Bnd. Another issue could be the requirement of $\mathsf{Bnd}(\xi) = \mathsf{Bnd}(\eta)$ for all action formulae $\xi \vee \eta$. This requirement forbids for instance the formula $l : \mathbf{newloc}\ u_1 \vee l : \mathbf{newloc}\ u_2$. A possible alternative could be to drop the requirement, defining $\mathsf{Bnd}(\xi \vee \eta) \stackrel{\text{def}}{=}$ if $(\mathsf{Bnd}(\xi) \neq \varnothing) \wedge (\mathsf{Bnd}(\eta) \neq$

$\varnothing$) then $(\mathsf{Bnd}(\xi) \cup \mathsf{Bnd}(\eta))$ else $\varnothing$. At the semantics level, when the formula is matched against $l : \mathbf{newloc}\ l'$ the substitution $[l'/u_1, l'/u_2]$ will be generated. On the other hand, one should still take care that formulae like $l_1 : \mathbf{newloc}\ u_1 \vee l_2 : \mathbf{newloc}\ u_2$ are not allowed since whatever substitution will be generated at the semantics level, this would bind at most one of u_1 or u_2. In the present paper we adopt simpler, but safe solutions for both issues.

3.2 Semantics of *MoSL*

MoSL state formulae are interpreted over the states of $AMC(N, \beta) = (S, Act, \mapsto)$ obtained from a StocKlaim network specification (β, N) as described in Sect. 2. We recall here that the states of such an AMC are StocKlaim configurations. *MoSL* formulae may contain locality variables which are instantiated by proper *substitutions* as described in the definition of the satisfaction relation below. Such substitutions are generated whenever an action $(l, \mathbf{newloc}\ l')$ in a path of $AMC(N, \beta)$ satisfies an action formula of the form $l : \mathbf{newloc}\ u$, for some $u \in \mathcal{U}$, and the associated substitution is $[l'/u]$. As usual, by $\Phi[l_1/u_1, \ldots, l_n/u_n]$ we denote the formula where all free locality variables of Φ belonging to $\{u_1, \ldots, u_n\}$ are replaced by the corresponding localities $l_1, \ldots, l_n$. We let [] denote the empty substitution and, without loss of generality, for $\Theta_1 = [l_1/u_1, \ldots, l_n/u_n, l'_1/u'_1, \ldots, l'_m/u'_m]$ and $\Theta_2 = [l''_1/u'_1, \ldots, l''_m/u'_m, l''_{m+1}/u'_{m+1}, \ldots, l''_{m+h}/u'_{m+h}]$, with $\{u'_{m+1}, \ldots, u'_{m+h}\} \cap \{u_1, \ldots, u_n\} = \varnothing$ we let $\Theta_1 \triangleleft \Theta_2$ be defined as the substitution

$$[l_1/u_1, \ldots, l_n/u_n, l''_1/u'_1, \ldots, l''_m/u'_m, l''_{m+1}/u'_{m+1}, \ldots, l''_{m+h}/u'_{m+h}]$$

The satisfaction relation for *state-formulae* is given in Table 4 and is self-explanatory. The def-

$s \models \mathrm{tt}$	$s \models A@l$ iff $N_s \equiv N' \parallel l :: A$
$s \models \neg\Phi$ iff not $s \models \Phi$	$s \models \langle l' \rangle @l$ iff $N_s \equiv N' \parallel l :: \langle l' \rangle$
$s \models \Phi \vee \Psi$ iff $s \models \Phi$ or $s \models \Psi$	$s \models \langle A \rangle @l$ iff $N_s \equiv N' \parallel l :: \langle A \rangle$
$s \models \mathcal{S}_{\bowtie p}(\Phi)$ iff $\lim_{t\to\infty} \Pr\{\sigma \in Paths(s) \mid \sigma(t) \models \Phi\} \bowtie p$	
$s \models \mathcal{P}_{\bowtie p}(\varphi)$ iff $\Pr\{\sigma \in Paths(s) \mid \sigma \models \varphi\} \bowtie p$	

Table 4. Satisfaction relation for state-formulae.

inition of the satisfaction relation for *path-formulae* given in Table 5 makes use of the substitution generator function SBS, where $\mathsf{SBS}((l, \mathbf{newloc}\ l'), l : \mathbf{newloc}\ u) \stackrel{\text{def}}{=} [l'/u]$, $\mathsf{SBS}(\gamma, \xi \vee \eta) \stackrel{\text{def}}{=} \mathsf{SBS}(\gamma, \xi) \triangleleft \mathsf{SBS}(\gamma, \eta)$, and $\mathsf{SBS}(\gamma, \xi) \stackrel{\text{def}}{=} []$ in all other cases. Note that the the well-formedness of (action) formulae guarantees that SBS generates a substitution, which binds at most one locality variable. As it should be clear from the formal definition, as soon as an action in a path satisfies an action formula of the form $\mathbf{newloc}\ u$, variable u is properly bound and the substitution thus generated is used for binding the free variables in the scope of the action formula. The interpretation of action formulae is given in Table 6.

The operators of *MoSL* can be used for defining some useful and frequently used derived operators (see Table 7).

4 A translation of *MoSL* to aCSL

In this section we present a translation from a large fragment of *MoSL* to aCSL and we show its correctness. The fragment includes all *MoSL* formulae except those which contain sub formulae of the form $\ell : \mathbf{newloc}\ u$. Our conjecture is that the latter formulae can be represented as proper disjunctions indexed with all possible values u can take for a given AMC, as we shall briefly discuss at the end of the present section. In the following, we let $MoSL^-$ denote the restricted language.

$\sigma \models \Phi\, {}_{\xi}\mathbf{U}^{<t}\, \Psi$ iff $\sigma[0] \models \Psi$ or there exists $k : 0 < k \leqslant (\mathsf{len}\sigma)$ such that
the following three conditions hold:
- $\sigma[k] \models \Psi\Theta_k$
- $t > \sum_{i=0}^{k-1} \delta(\sigma, i)$
- $\sigma[i] \models \Phi\Theta_i$ and $\alpha(\sigma, i) \models \xi\Theta_i$, for all i with $0 \leqslant i < k$

where $\Theta_0 \stackrel{\text{def}}{=} []$
$\Theta_i \stackrel{\text{def}}{=} \Theta_{i-1} \triangleleft \mathsf{SBS}(\alpha(\sigma, i-1), \xi)$ for all $i > 0$

$\sigma \models \Phi\, {}_{\xi}\mathbf{U}_{\eta}^{<t}\, \Psi$ iff there exists $k : 0 < k \leqslant (\mathsf{len}\sigma)$ such that
the following four conditions hold:
- $\sigma[k] \models \Psi\Theta_k$
- $t > \sum_{i=0}^{k-1} \delta(\sigma, i)$
- $\sigma[k-1] \models \Phi\Theta_{k-1}$ and $\alpha(\sigma, k-1) \models \eta\Theta_{k-1}$
- $\sigma[i] \models \Phi\Theta_i$ and $\alpha(\sigma, i) \models \xi\Theta_i$, for all i with $0 \leqslant i < k-1$

where $\Theta_0 \stackrel{\text{def}}{=} []$
$\Theta_k \stackrel{\text{def}}{=} \Theta_{k-1} \triangleleft \mathsf{SBS}(\alpha(\sigma, k-1), \eta)$
$\Theta_i \stackrel{\text{def}}{=} \Theta_{i-1} \triangleleft \mathsf{SBS}(\alpha(\sigma, i-1), \xi)$ for all i with $0 < i < k$

Table 5. Satisfaction relation for path-formulae.

$\gamma \models \mathrm{tt}$		$\gamma \models l : \mathbf{in}\ l' @ l''$	iff $\gamma = (l, \mathbf{in}\ l' @ l'')$
$\gamma \models \neg\, \xi$	iff not $\gamma \models \xi$	$\gamma \models l : \mathbf{in}\ A @ l'$	iff $\gamma = (l, \mathbf{in}\ A @ l')$
$\gamma \models \xi \vee \eta$	iff $\gamma \models \xi$ or $\gamma \models \eta$	$\gamma \models l : \mathbf{eval}\ A @ l'$	iff $\gamma = (l, \mathbf{eval}\ A @ l')$
$\gamma \models l : \mathbf{out}\ l' @ l''$	iff $\gamma = (l, \mathbf{out}\ l' @ l'')$	$\gamma \models l : \mathbf{newloc}\ u$	iff there exists $l' \in \mathcal{L}$
$\gamma \models l : \mathbf{out}\ A @ l'$	iff $\gamma = (l, \mathbf{out}\ A @ l')$		such that $\gamma = (l, \mathbf{newloc}\ l')$

Table 6. Satisfaction relation for action-formulae.

Given a $MoSL^{-}$ formula $\hat{\Phi}$, and a network specification (N, β), and assuming its LTS $TS(N, \beta) = (C, \Lambda, \rightarrow, c_0)$ finite, with $AMC(N, \beta) = (S, Act, \mapsto)$ being the related AMC, the question is how to translate $\hat{\Phi}$ into an aCSL formula in order to perform model checking using an existing aCSL model-checker.

Since in aCSL only *action* atomic propositions can be expressed, the first step of our procedure is concerned with finding a way for incorporating *state* atomic propositions into the transition labels of the AMC. To that purpose, let $\aleph_1, \ldots, \aleph_n$ be *all* the atomic propositions occurring in $\hat{\Phi}$. For notational simplicity, we associate a unique name p_j to each $\aleph_j$ above. Moreover, we use such names for building factors, in boolean product expressions, of the form $z_1 \ldots z_n \in B(p_1, \ldots, p_n) \stackrel{\text{def}}{=} \times_{j=1}^{n} \{p_j, \bar{p_j}\}$, with $p_i \neq p_j$ for $i \neq j$. For instance, $B(p_1, p_2)$ is the set $\{\bar{p_1}, \bar{p_2}, \bar{p_1}p_2, p_1\bar{p_2}, p_1p_2\}$.

We define the *characteristic function* of S as the function χ from S to $B(p_1, \ldots, p_n)$ such that, for all $s \in S$, $\chi(s) \stackrel{\text{def}}{=} z_1 \ldots z_n$ such that, for $j = 1, \ldots, n$, $z_j = p_j$ if $s \models \aleph_j$ and $z_j = \bar{p_j}$ if

$\Phi\, {}_{\xi}\mathbf{U}_{\eta}\, \Psi$	$\stackrel{\text{def}}{=}\ \Phi\, {}_{\xi}\mathbf{U}_{\eta}^{<\infty}\, \Psi$	$\mathcal{P}_{\bowtie p}({}_{A}\Diamond_{B}^{<t}\ \Phi)$	$\stackrel{\text{def}}{=}\ \mathcal{P}_{\bowtie p}(\mathrm{tt}\ {}_{A}\mathbf{U}_{B}^{<t}\, \Phi)$
$\Phi\, \mathbf{U}\, \Psi$	$\stackrel{\text{def}}{=}\ \Phi\, {}_{\mathrm{tt}}\mathbf{U}\, \Psi$	$\mathcal{P}_{\bowtie p}({}_{A}\Box_{B}^{<t}\ \Phi)$	$\stackrel{\text{def}}{=}\ \neg\mathcal{P}_{\bowtie p}({}_{A}\Diamond_{B}^{<t}\ \neg\Phi)$
$\mathbf{X}_{\xi}^{<t}\, \Phi$	$\stackrel{\text{def}}{=}\ \mathrm{tt}\, {}_{\mathrm{ff}}\mathbf{U}_{\xi}^{<t}\, \Phi$	$\mathcal{P}_{\bowtie p}({}_{A}\Diamond^{<t}\ \Phi)$	$\stackrel{\text{def}}{=}\ \mathcal{P}_{\bowtie p}(\mathrm{tt}\ {}_{A}\mathbf{U}^{<t}\, \Phi)$
$\langle\xi\rangle\, \Phi$	$\stackrel{\text{def}}{=}\ \mathcal{P}_{>0}(\mathbf{X}_{\xi}\, \Phi)$	$\mathcal{P}_{\bowtie p}({}_{A}\Box^{<t}\ \Phi)$	$\stackrel{\text{def}}{=}\ \neg\mathcal{P}_{\bowtie p}({}_{A}\Diamond^{<t}\ \neg\Phi)$
$[\xi]\, \Phi$	$\stackrel{\text{def}}{=}\ \neg\langle\xi\rangle\neg\Phi$		

Table 7. Derived operators.

$s \models \aleph_j$ does not hold. Notice that the above satisfiability check is computed by a simple (static) analysis of (the network component of) s. The idea behind this representation is that we take $B(p_1, \ldots, p_n)$ as the domain for an interpretation function R over $MoSL^-$ in such a way that each atomic proposition—or boolean combination of atomic propositions—is mapped into the (set of the disjuncts of the) sum-product-form expression representing it. Formulas containing probabilistic/temporal operators are instead mapped into the (set of disjuncts of the) sum-product-form expression of true (tt). Function R is then used for building, by means of function A, proper action index-sets (in the until sub-formulae) of the aCSL formula $\mathsf{T}(\hat{\Phi})$ resulting from the translation of $\hat{\Phi}$. Before giving the formal definition of the above mentioned functions, we define the associated translation on the AMC side.

We transform $AMC(N, \beta)$ into another AMC, $FAMC(N, \beta)$, by moving the relevant state information *forward* to the transitions emanating from states.

Definition 5. *Given AMC* $AMC(N, \beta) = (S, Act, \mapsto)$ *and bijective encoding* cod *of* $Act \times B(p_1, \ldots, p_n)$ *into finite set* Υ*, we define* $FAMC(N, \beta)$ *as the AMC* $(S_F, Act_F, \mapsto_F)$ *with* $S_F = S$*,* $Act_F \subseteq \Upsilon$ *and* $\mapsto_F$ *such that* $s \stackrel{\mathsf{cod}(\gamma, \chi(s)), \lambda}{\longmapsto_F} s'$ *if and only if* $s \stackrel{\gamma, \lambda}{\longmapsto} s'$.

For every path $\sigma \in \mathsf{Paths}(s)$ over $AMC(N, \beta)$ there is a corresponding path σ_F over $FAMC(N, \beta)$ where the obvious correspondence is the following one: for all $i \geqslant 0$, $\sigma_F[i] = \sigma[i]$, $\alpha(\sigma_F, i) = \mathsf{cod}(\alpha(\sigma, i), \chi(\sigma[i+1]))$, and $\delta(\sigma_F, i) = \delta(\sigma, i)$. We let $\mathsf{Paths}_F(s)$ denote the set of all such paths.

It is worth pointing out that this transformation cannot deal properly with *absorbing* states. In the following we assume $AMC(N, \beta)$ does not contain absorbing states (they can be eliminated by equipping them with proper self-loops).

The translation function T is defined in Fig. 2. It uses function A which is defined as follows

$$A(\xi, \Phi) \stackrel{\text{def}}{=} \{\mathsf{cod}(\gamma, p) \in Act_F \mid \gamma \in \mathsf{set}(\xi), p \in R(\Phi)\}$$

where $\mathsf{set}(\xi)$ is the set of actions corresponding to action formula ξ; its definition is left out here, being the obvious one (e.g. $\mathsf{set}(l_1 : \mathbf{out}\ l_2\ @\ l_3 \vee l_4 : \mathbf{in}\ l_5\ @\ l_6) = \{(l_1, \mathbf{out}\ l_2\ @\ l_3), (l_4, \mathbf{in}\ l_5\ @\ l_6)\}$). It is easy to see that the following Lemma holds:

Lemma 1.
For all $N, \beta, s, \Phi, \gamma, \xi$*, the following holds:* $\mathsf{cod}(\gamma, \chi(s)) \in A(\xi, \Phi)$ *implies* $AMC(N, \beta), \gamma \models \xi$ □

Function R is defined in Fig. 3 and, as we anticipated above, for each atomic state proposition—or boolean combinations of atomic propositions—it generates the associated representation in $B(p_1, \ldots, p_n)$. For instance, assuming $\aleph_j$ represented by p_j, for $j = 1, 2$ we have $R(\neg \mathrm{tt}) = \varnothing$, $R(\mathrm{tt}) = \{\bar{p_1}\bar{p_2}, \bar{p_1}p_2, p_1\bar{p_2}, p_1p_2\}$, and $R(\aleph_1) = R(\neg\aleph_2) = \{p_1\bar{p_2}, p_1p_2\}$ On formulae containing also stochastic or temporal operators R behaves as for tt. Predicate PCF characterizes the subset of $MoSL^-$ formulae which do not contain modal operators, i.e. formulae which are only (boolean combinations of) state-atomic propositions.

Function R enjoys the property stated by Lemma 2 below, which will be exploited in the proof of correctness for the translation procedure[2].

Lemma 2.
For all $N, \beta, s,$ *and* Φ *the following holds:*

i) if $\mathsf{PCF}(\Phi)$ *then we have* $AMC(N, \beta), s \models \Phi$ *iff* $\chi(s) \in R(\Phi)$
ii) $AMC(N, \beta), s \models \Phi$ *implies* $\chi(s) \in R(\Phi)$ □

An immediate consequence of Lemma 2, and of the definitions of functions A and cod is the following Lemma:

Lemma 3.
For all $N, \beta, s, \Phi, \gamma, \xi$ *such that* $AMC(N, \beta), \gamma \models \xi$*, the following holds:* $AMC(N, \beta), s \models \Phi$ *implies* $\mathsf{cod}(\gamma, \chi(s)) \in A(\xi, \Phi)$ □

$$
\begin{array}{ll}
\mathsf{T}(\mathrm{tt}) & \stackrel{\text{def}}{=} \mathrm{tt} \\
\mathsf{T}(\aleph) & \stackrel{\text{def}}{=} \mathcal{P}_{\geqslant 1}(\mathbf{X}_{A(\mathrm{tt},\aleph)}\ \mathrm{tt}) \\
\mathsf{T}(\neg \Phi) & \stackrel{\text{def}}{=} \neg \mathsf{T}(\Phi) \\
\mathsf{T}(\Phi \vee \Phi') & \stackrel{\text{def}}{=} \mathsf{T}(\Phi) \vee \mathsf{T}(\Phi')
\end{array}
\qquad
\begin{array}{ll}
\mathsf{T}(\mathcal{S}_{\bowtie p}(\Phi)) & \stackrel{\text{def}}{=} \mathcal{S}_{\bowtie p}(\mathsf{T}(\Phi)) \\
\mathsf{T}(\mathcal{P}_{\bowtie p}(\varphi)) & \stackrel{\text{def}}{=} \mathcal{P}_{\bowtie p}(\mathsf{T}(\varphi)) \\
\mathsf{T}(\Phi\ {}_{\xi}\mathbf{U}^{<t}\ \Phi') & \stackrel{\text{def}}{=} \mathsf{T}(\Phi)\ {}_{A(\xi,\Phi)}\mathbf{U}^{<t}\ \mathsf{T}(\Phi') \\
\mathsf{T}(\Phi\ {}_{\xi}\mathbf{U}^{<t}_{\xi'}\ \Phi') & \stackrel{\text{def}}{=} \mathsf{T}(\Phi)\ {}_{A(\xi,\Phi)}\mathbf{U}^{<t}_{A(\xi',\Phi)}\ \mathsf{T}(\Phi')
\end{array}
$$

Fig. 2. Logic Translation function

Function T essentially moves every requirement on states $\aleph$ which is an atomic proposition—or a boolean combination thereof—to a requirement on *all* transitions emanating from such states. Intuitively, it is required that the labels of such transitions are "marked" by the requirement. Technically, this is achieved by requiring them be elements of $A(\mathrm{tt}, \aleph)$ and it is the logic counterpart of the definition of $\mapsto_F$. Notice that we use the fact that the aCSL path operator $\mathcal{P}_{\geqslant 1}(\cdot)$ expresses the CTL path-quantifier $\forall\, \cdot$. Such correspondence is justified only under specific fairness conditions [3]. On the other hand, the specific form of formulae we are dealing with (i.e. $\mathbf{X}_{A(\mathrm{tt},\aleph)}\ \mathrm{tt}$), together with the fact that, by construction of $FAMC(N, \beta)$, either all transitions emanating from a state are included in $A(\mathrm{tt}, \aleph)$ or none of them is, make the fairness constraints irrelevant for the case at hand. The only other interesting cases of the definition of T are those for the until formulae. Notice that action requirements ξ and ξ' are enriched with those coming from formula Φ. In particular, this holds for ξ' due to the fact that state properties are moved *forward* to emanating transitions.

$$
\begin{array}{ll}
R(\mathrm{tt}) & \stackrel{\text{def}}{=} B(p_1, \ldots, p_n) \\
R(\aleph_j) & \stackrel{\text{def}}{=} \{z_1 \ldots z_n \mid z_j = p_j, z_i \in \{p_i, \bar{p}_i\}, \\
 & \quad \text{for } j \neq i = 1, \ldots, n\},\ \text{for } j = 1, \ldots, n \\
R(\neg \Phi) & \stackrel{\text{def}}{=} R(\mathrm{tt}) \setminus R(\Phi),\ \text{if}\ \mathsf{PCF}(\neg \Phi) \\
 & \stackrel{\text{def}}{=} R(\mathrm{tt}),\ \text{otherwise} \\
\text{where} & \\
\mathsf{PCF}(\mathrm{tt}) & \stackrel{\text{def}}{=} \mathrm{tt} \\
\mathsf{PCF}(\aleph_j) & \stackrel{\text{def}}{=} \mathrm{tt},\ \text{for } j = 1, \ldots, n \\
\mathsf{PCF}(\neg \Phi) & \stackrel{\text{def}}{=} \mathsf{PCF}(\Phi)
\end{array}
\qquad
\begin{array}{ll}
R(\Phi \vee \Phi') & \stackrel{\text{def}}{=} R(\Phi) \cup R(\Phi'),\ \text{if}\ \mathsf{PCF}(\Phi \vee \Phi') \\
 & \stackrel{\text{def}}{=} R(\mathrm{tt}),\ \text{otherwise} \\
R(\mathcal{S}_{\bowtie p}(\Phi)) & \stackrel{\text{def}}{=} R(\mathrm{tt}) \\
R(\mathcal{P}_{\bowtie p}(\varphi)) & \stackrel{\text{def}}{=} R(\mathrm{tt}) \\
 & \\
 & \\
\mathsf{PCF}(\Phi \vee \Phi') & \stackrel{\text{def}}{=} \mathsf{PCF}(\Phi) \wedge \mathsf{PCF}(\Phi') \\
\mathsf{PCF}(\mathcal{S}_{\bowtie p}(\Phi)) & \stackrel{\text{def}}{=} \mathrm{ff} \\
\mathsf{PCF}(\mathcal{P}_{\bowtie p}(\varphi)) & \stackrel{\text{def}}{=} \mathrm{ff}
\end{array}
$$

Fig. 3. Function R

The three lemmas above are used in the proof of the main theorem below:

Theorem 1.
For $MoSL^-$ formula Φ, and a network specification (N, β), the following holds: for all states s of $AMC(N, \beta)$, $AMC(N, \beta), s \models_{MoSL^-} \Phi$ iff $FAMC(N, \beta), s \models_{aCSL} \mathsf{T}(\Phi)$. □

5 Modeling and analysis of the spreading of a virus

In this section we show how StocKlaim can be used for modeling the *spreading* of a virus in a network. We also give examples of interesting qualitative and quantitative properties of the model that can be expressed in $MoSL$. This example has been inspired by a similar one in [16] and used also in [13].

[2] In the sequel, in order to avoid confusion, we will explicitly indicate the AMC in the satisfiability relation: e.g. $\mathcal{M}, s \models \Phi$ indicates that $s \models \Phi$ in AMC $\mathcal{M}$.

We model a network as a set of nodes and the virus running on a node can move arbitrarily from the current node to a subset of adjacent nodes, infecting them. At each node, an operating system runs, which upon receiving the virus, can either run it or suppress it. In this paper, for the sake of simplicity we consider simple networks which are in fact grids of $n \times m$ nodes. Each node is connected with its four neighbors (north, south, east, west), except for border nodes, which lack some connections in the obvious way (e.g. the nodes on the east border have no east connection). Moreover, we assume that the virus can move only to *one* adjacent node. Finally, we refrain from modeling aspects of the virus other than the way it replicates in the network. In particular we do not consider the local effects of the virus and we make the virus die as soon as it has infected one of the neighbors of its locality.

The specification schema of the virus and the operating system running at each node is given in Fig. 4, where a network is conventionally represented as a $n \times m$ matrix of localities l_{ij}.

$\mathtt{V}_{ij} \triangleq$ (**out** $\mathtt{V}_{i-1j}$ @ $\mathtt{l}_{i-1j}, \mathtt{n}_{ij}$).**nil** +
/* alternative present only for $i > 1$ */
(**out** $\mathtt{V}_{i+1j}$ @ $\mathtt{l}_{i+1j}, \mathtt{s}_{ij}$).**nil** +
/* alternative present only for $i < n$ */
(**out** $\mathtt{V}_{ij+1}$ @ $\mathtt{l}_{ij+1}, \mathtt{e}_{ij}$).**nil** +
/* alternative present only for $j < m$ */
(**out** $\mathtt{V}_{ij-1}$ @ $\mathtt{l}_{ij-1}, \mathtt{w}_{ij}$).**nil**
/* alternative present only for $j > 1$ */

$\mathtt{O}_{ij} \triangleq$ (**in** !C @ $\mathtt{l}_{ij}, \mathtt{u}_{ij}$).$\mathtt{X}_{ij}$ +
/* the received virus is undetected and will run */
(**in** !C @ $\mathtt{l}_{ij}, \mathtt{d}_{ij}$).$\mathtt{O}_{ij}$
/* the received virus is detected and suppressed */

$\mathtt{X}_{ij} \triangleq$ (**eval** C @ $\mathtt{l}_{ij}, \mathtt{r}_{ij}$).$\mathtt{O}_{ij}$
/* the virus is activated */

Fig. 4. Specification of an infected network

For the verification, we chose $n = m = 3$ with the following initial state N_0: $\mathtt{l}_{11}::\mathtt{O}_{11} \parallel \mathtt{l}_{11}::\langle \mathtt{V}_{11} \rangle$, while $\mathtt{l}_{ij}::\mathtt{O}_{ij}$ for $1 \leqslant i, j \leqslant 3$ with $i \neq 1$ or $j \neq 1$. The resulting LTS is not shown for space reasons; it consists of 28 states and 52 transitions.

There are several interesting issues of the spreading of the virus which can be addressed using *MoSL*. The first property, Φ_1, is an example of a purely state-based quantitative property. The probability that the virus is running at node $\mathtt{l}_{ij}$ within t time-units after the infection of node $\mathtt{l}_{11}$ is smaller than a given upper bound p. This property becomes more interesting when we define the rates associated to the detection (resp. lack of detection) of the virus in such a way that the operating systems of the localities on the diagonal from bottom-left to top-right—$\mathtt{O}_{31}$, $\mathtt{O}_{22}$, and $\mathtt{O}_{13}$—have a relatively high rate of detection and can be considered as a firewall to protect the nodes $\mathtt{l}_{32}$, $\mathtt{l}_{33}$, and $\mathtt{l}_{23}$.

The property can be expressed in *MoSL* for locality $\mathtt{l}_{33}$ and $p = 0.2$ as follows:

$$\mathcal{P}_{\leqslant 0.2}(\neg(\mathtt{V}_{33}@\mathtt{l}_{33})\ \mathrm{tt}\ \mathbf{U}^{\leqslant t}\ \mathtt{V}_{33}@\mathtt{l}_{33})$$

Let $\aleph_1$ stand for $\mathtt{V}_{33}@\mathtt{l}_{33}$ and assume it be represented by q. Let also $VrsAct$ be the set of actions of $AMC(N_0, \beta_0)$ and assume the encoding be defined simply as [3] $\mathsf{cod}(\gamma, z) \stackrel{\text{def}}{=} (\gamma, z)$. The translation $\mathsf{T}(\Phi_1)$ then yields the following aCSL formula:

$$\mathcal{P}_{\leqslant 0.2}(\neg\mathcal{P}_{\geqslant 1}(\mathbf{X}_{A(\mathrm{tt},\aleph_1)}\ \mathrm{tt})\ _{A(\mathrm{tt},\neg\aleph_1)}\mathbf{U}^{\leqslant t}\ \mathcal{P}_{\geqslant 1}(\mathbf{X}_{A(\mathrm{tt},\aleph_1)}\ \mathrm{tt}))$$

[3] In practice, due to lexical restrictions on action labels imposed by the implementation of ETMCC, the encodings we used in actual experiments are slightly more involving.

where $A(\mathrm{tt}, \aleph_1)$ is the set $\{(\gamma, q) \in VrsAct_F \mid \gamma \in VrsAct\}$ and, similarly, $A(\mathrm{tt}, \neg\aleph_1)$ is the set $\{(\gamma, \bar{q}) \in VrsAct_F \mid \gamma \in VrsAct\}$. Fig. 5 shows the probability to reach, from the initial state, a state where the virus is running in locality $\mathtt{l}_{33}$. The measure is presented for time values ranging from 1 to 10 with $\beta_0\, e_{ij} = \beta_0\, n_{ij} = \beta_0\, s_{ij} = \beta_0\, w_{ij} = \beta_0\, r_{ij} = 2$ for $1 \leqslant i, j \leqslant 3$, $\beta_0\, d_{31} = \beta_0\, d_{22} = \beta_0\, d_{13} = 10$, and $\beta_0\, d_{ij} = 1$ otherwise, $\beta_0\, u_{31} = \beta_0\, u_{22} = \beta_0\, u_{13} = 1$, and $\beta_0\, u_{ij} = 10$ otherwise.

We performed similar analyzes for different values of the detection (resp. lack of detection) rates of the firewall. In particular for d_{31}, d_{22}, d_{13} and u_{31}, u_{22}, u_{13} ranging over $[1, \ldots, 10]$, with $d_{(4-i)i} + u_{(4-i)i}$ constant for $1 \leqslant i \leqslant 3$ (and equal to 11).

For the sake of readability, in Fig. 5 we show the results only for $d_{31}, d_{22}, d_{13} \in \{1, 6, 10\}$ and $u_{31}, u_{22}, u_{13} \in \{1, 5, 10\}$. The results clearly indicate that for high detection rates the probability

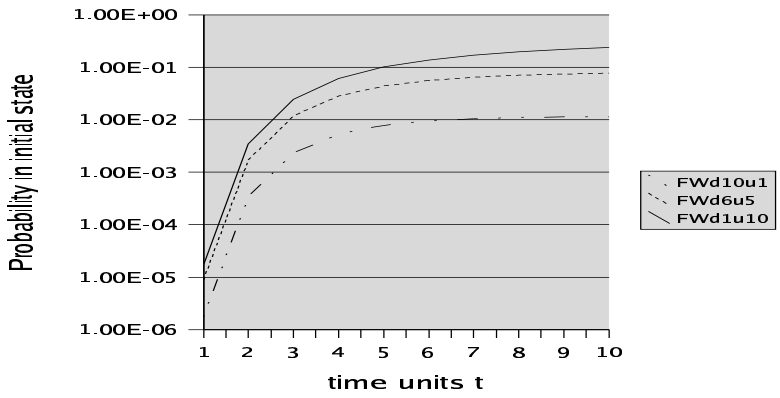

Fig. 5. Results for Firewalls with different detection capability

for locality l_{33} to run the virus within a certain time interval is lower.

Stochastic model-checking permits also the verification of qualitative properties as a degenerate case of quantitative ones. For instance, an interesting property is: "whenever a node is infected (i.e. a virus runs on it) the virus may move to a neighbor in the next step". For instance, in the case of node $\mathtt{l}_{33}$ the property of interest, Φ_2, is:

$$\mathtt{V}_{33}@\mathtt{l}_{33} \Rightarrow \langle \mathtt{l}_{33} : \mathbf{out}\ \mathtt{V}_{32}\ @\ \mathtt{l}_{32} \rangle\, \mathrm{tt}$$

The translated formula $\mathsf{T}(\Phi_2)$ is given below, where we let ξ stand for $\mathtt{l}_{33} : \mathbf{out}\ \mathtt{V}_{32}\ @\ \mathtt{l}_{32}$, and $\aleph_1$ as before:

$$(\neg \mathcal{P}_{\geqslant 1}(\mathbf{X}_{A(\mathrm{tt},\aleph_1)}\, \mathrm{tt})) \vee (\mathcal{P}_{>0}(\mathrm{tt}\ \mathscr{U}_{A(\xi,\mathrm{tt})}\, \mathrm{tt}))$$

Set $A(\xi, \mathrm{tt})$ is the set $\{(\mathtt{l}_{33} : \mathbf{out}\ \mathtt{V}_{32}\ @\ \mathtt{l}_{32}, q), (\mathtt{l}_{33} : \mathbf{out}\ \mathtt{V}_{32}\ @\ \mathtt{l}_{32}, \bar{q})\} \cap VrsAct_F$. The model-checker shows that Φ_2 holds in every state.

6 Conclusions

In this paper *MoSL*, a stochastic logic for StocKlaim has been proposed. StocKlaim is a stochastic extension of the core subset of Klaim, a prototype language for modelling and programming global or network-aware computers. StocKlaim addresses process mobility, process distribution, asynchronous communication through shared local repositories (i.e., tuple spaces), and dynamic node creation, as well as randomly delayed activities.

The logic addresses both spatial and temporal notions to reflect both the topological structure of systems and their evolution over time. In connection with the duration attributes of StocKlaim

process actions, the logic provides probabilistic operators which naturally express steady-state probabilities as well as probability measures of paths specified with typical until formulae. The logic integrates both the state-based paradigm and the action-based one and provides specific state atomic propositions addressing data and process distribution. It also provides specific atomic propositions for actions in order to characterize relevant activities taking place during executions.

The formal semantics of $MoSL$ has been presented and a mapping from a large fragment of the logic to aCSL, the action based Continuous Stochastic Logic described in [23], has been formally defined and shown correct. The availability of such mapping(s) provides the possibility of model-checking systems modelled by StocKlaim against requirements specified in $MoSL$ using existing model-checkers for aCSL, like ETMCC. Together with M. Loreti, we are currently implementing the translation function between the two logics and the actual associations of a transition system to StocKlaim processes that will represent the model for the formulae to be checked.

The next research steps we intend to take are on one hand the implementation of the above mentioned mapping, and on the other to investigate the possibility of extending ξ to deal also with formulae containing **newloc** . The first attempt we shall make is to assume existence of a finite number of locations and to model location creation as a nondeterministic choice between those locations that are still unused. The translation would follow the pattern below:

If we let AMC be system $\mathcal{M} = (S, Act, \mapsto)$, and let $\mathsf{NewLocs}(\mathcal{M})$ be defined as the set of fresh locations:

$$\{l \mid \exists c, c', l', \lambda.\ c \xmapsto{(l', \mathbf{newloc}\ l), \lambda} c'\}$$

we can define $\mathsf{T}(\Phi_{l_1:\mathbf{newloc}\ u_1} \mathbf{U}^{<t}_{l_2:\mathbf{newloc}\ u_2} \Phi')$ as follows:

$$\bigvee_{l'_1, l'_2 \in \mathsf{NewLocs}(AMC(N,\beta))} \mathsf{T}(\Phi[l'_1/u_1])\ _{A(l_1:\mathbf{newloc}\ l'_1, \Phi[l'_1/u_1])} \mathbf{U}^{<t}_{A(l_2:\mathbf{newloc}\ l'_2, \Phi'[l'_1/u_1, l'_2/u_2])}\ \mathsf{T}(\Phi'[l'_1/u_1, l'_2/u_2])$$

Together with the implementation of the translations that would guarantee a rapid prototyping of a model checker for $MoSL$ we shall also investigate feasibility and convenience to develop direct model-checking algorithm. Another issue will be the extension of StocKlaim in order to cover a larger subset of Klaim and the related extension of $MoSL$.

We shall also consider more expressive logic that allows to reason about (spontaneous) sites failures and shall study paradigmatic examples to assess adequacy and expressiveness of the proposed logics. In this respect, we shall also investigate the relationships between behavioral equivalences (e.g. those bisimulation based) and the equivalences induced by the considered logics.

References

1. A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model checking Continuous Time Markov Chains. *ACM Transactions on Computational Logic*, 1(1):162–170, 2000.
2. C. Baier, J. Katoen, and H. Hermanns. Approximate Symbolic Model Checking of Continuous-Time Markov Chains. In J. Baeten and S. Mauw, editors, *Concur '99*, volume 1664 of *LNCS*, pages 146–162. Springer-Verlag, 1999.
3. C. Baier and M. Kwiatkowska. On the Verification of Qualitative Properties of Probabilistic Processes under Fairness Constraints. *Information Processing Letters*, 66(2):71–79, 1998.
4. L. Bettini, R. De Nicola, and M. Loreti. Formalizing Properties of Mobile Agent Systems. In F. Arbab and C. Talcott, editors, *Coordination Models and Languages*, volume 2315 of *LNCS*, pages 72–87. Springer-Verlag, 2002.
5. L. Bettini, V. Non, R. De Nicola, G. Ferrari, D. Gorla, M. Loreti, E. Moggi, R. Pugliese, E. Tuosto, and B. Venneri. The Klaim Project: Theory and Practice. In C. Priami, editor, *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, volume 2874 of *LNCS*, pages 88–150. Springer-Verlag, 2003.
6. L Caires and L. Cardelli. A spatial logic for concurrency (part I). *Information and Computation. Academic Press, Inc.*, 186(2):194–235, 2003.
7. L. Cardelli. A Language with Distributed Scope. In *22nd Annual ACM Symposium on Principles of Programming Languages*, pages 286–297. ACM, 1995.

8. L. Cardelli. Abstractions for Mobile Computations. In J. Vitek and C. Jensen, editors, *Secure Internet Programming*, volume 1603 of *LNCS*, pages 51–94. Springer-Verlag, 1999.
9. L. Cardelli and A. Gordon. Anytime, anywhere: modal logics for mobile ambients. In *Twentyseventh Annual ACM Symposium on Principles of Programming Languages*, pages 365–377. ACM, 2000.
10. G. Castagna and J. Vitek. Seal: A framework for Secure Mobile Computations. In H. Bal, B. Belkhouche, and L. Cardelli, editors, *Internet Programming Languages*, volume 1686 of *LNCS*, pages 47–77. Springer-Verlag, 1999.
11. P. D'Argenio, J. Katoen, and E. Brinksma. Specification and analysis of soft real-time systems: Quantity and quality. In *Real-Time Systems Symposium*, pages 104–114. IEEE - The Institute of Electrical and Electronic Engineers, 1999.
12. R. De Nicola, G. Ferrari, and R. Pugliese. KLAIM: A Kernel Language for Agents Interaction and Mobility. *IEEE Transactions on Software Engineering. IEEE CS*, 24(5):315–329, 1998.
13. R. De Nicola, D. Latella, and M. Massink. Formal modeling and quantitative analysis of KLAIM-based mobile systems. In *Proceedings of the 20th Annual ACM Symposium on Applied Computing - Special Track on Coordination Models, Languages and Applications*. Association for Computing Machinery - ACM, 2005. (To appear).
14. R. De Nicola and M. Loreti. A modal logic for mobile agents. *ACM Transactions on Computational Logic. ACM Press*, 5(1):79–128, 2004.
15. R. De Nicola and F. Vaandrager. Action versus state based logics for transition systems. In I. Guessarian, editor, *Proceedings of LITP Spring School on Theoretical Computer Science*, volume 469 of *LNCS*, pages 407–419. Springer-Verlag, 1990.
16. A. Di Pierro, C. Hankin, and H. Wiklicky. Probabilistic KLAIM. In R. De Nicola, G. Ferrari, and G. Meredith, editors, *Coordination Models and Languages*, volume 2949 of *LNCS*. Springer-Verlag, 2004.
17. A. Fantechi, S. Gnesi, and G. Mazzarini. How Much Expressive Are LOTOS Expressions? In J. Quemada, J. Manas, and M. Thomas, editors, *Formal Description Techniques — III*. North-Holland Publishing Company, 1991.
18. G. Ferrari, S. Gnesi, U. Montanari, and M. Pistore. A model-checking verification environment for mobile processes. *ACM Transactions on Software Engineering and Methodology. ACM Press*, 12(4):440–473, 2003.
19. D. Gorla and R. Pugliese. A Semantic Theory for Global Computing Systems, 2004. (Submitted for publication. Available at http://www.dsi.uniroma1.it/~gorla/papers/bis4k-full.pdf).
20. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing. The International Journal of Formal Methods. Springer-Verlag*, 6(5):512–535, 1994.
21. S. Hart and M. Sharir. Probabilistic Temporal Logics for Finite and Bounded Models. In R. De Millo, editor, *16th annual ACM symposium on Theory of computing*, pages 1–13. Association for Computing Machinery - ACM, 1984. ISBN 0-89791-133-4.
22. B. Haverkort. Markovian Models for Performance and Dependability Evaluation. In E. Brinksma, H. Hermanns, and J. Katoen, editors, *Lectures on Formal Methods and Performance Analysis*, volume 2090 of *LNCS*, pages 38–83. Springer-Verlag, 2001.
23. H. Hermanns, J. Katoen, J. Meyer-Kayser, and M. Siegle. Towards Model Checking Stochastic Process Algebra. In W. Grieskamp, T. Santen, and B. Stoddart, editors, *Integrated Formal Methods - IFM 2000*, volume 1945 of *LNCS*, pages 420–439. Springer-Verlag, 2000.
24. H. Hermanns, J. Katoen, J. Meyer-Kayser, and M. Siegle. A Tool for Model-Checking Markov Chains. *International Journal on Software Tools for Technology Transfer. Springer-Verlag*, 4(2):153–172, 2003.
25. V. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.
26. S. Merz, M. Wirsing, and J. Zappe. A Spatio-Temporal Logic for the Specification and Refinement of Mobile Systems. In M. Pezzé, editor, *Fundamental Approaches to Software Engineering (FASE 2003)*, volume 2621 of *LNCS*, pages 87–101. Springer-Verlag, 2003.

A Auxiliary definitions for StocKlaim Operational Semantics

Table 8 defines the Laws of the Structural Congruence on StocKlaim expressions which is used by the Reduction Rules of the operational semantics. Both the Laws and the Rules are designed in such a way that rate name uniqueness is preserved by their application, by means of function RN defined in Fig. 6.

$$L, \beta \vdash N_1 \parallel N_2 \equiv L, \beta \vdash N_2 \parallel N_1 \qquad (\mathrm{CO}\parallel)$$

$$L, \beta \vdash N_1 \parallel (N_2 \parallel N_3) \equiv L, \beta \vdash (N_1 \parallel N_2) \parallel N_3 \qquad (\mathrm{AS}\parallel)$$

$$L, \beta \vdash l :: P \equiv L, \beta \vdash l :: P \mid \mathbf{nil} \qquad (\mathrm{NE}|)$$

$$L, \beta \vdash l :: P_1 + P_2 \equiv L, \beta \vdash l :: P_2 + P_1 \qquad (\mathrm{CO}+)$$

$$L, \beta \vdash l :: P_1 + (P_2 + P_3) \equiv L, \beta \vdash l :: (P_1 + P_2) + P_3 \qquad (\mathrm{AS}+)$$

$$L, \beta \vdash l :: P \equiv L, \beta \vdash l :: P + \mathbf{nil} \qquad (\mathrm{NE}+)$$

$$L, \beta \vdash l :: P_1 \mid P_2 \equiv L, \beta \vdash l :: P_1 \parallel l :: P_2 \qquad (\mathrm{CLO})$$

$$L, \beta \vdash N \equiv L, (\beta \circ \theta^{-1}) \vdash N\theta \qquad (\mathrm{REN})$$
for any $\theta : \mathcal{R} \to \mathcal{R}$ injective, and such that
$dom\,\theta = (\mathsf{Rat}\,N)$ and $(rng\,\theta) \cap (dom\,\beta) = \varnothing$

Table 8. Structural Congruence of StocKlaim

$$\mathsf{RN}(\mathbf{nil}, \beta) \stackrel{\text{def}}{=} (\mathbf{nil}, \beta)$$

$$\mathsf{RN}((a, r).P, \beta) \stackrel{\text{def}}{=} ((a, r').P', \beta') \quad \text{where } r' = \mathbf{choose}\ r_1 \in \mathcal{R} \setminus (dom\,\beta),\ (P', \beta') = \mathsf{RN}(P, \beta[\beta(r)/r'])$$

$$\mathsf{RN}(P\ op\ Q, \beta) \stackrel{\text{def}}{=} (P'\ op\ Q', \beta'), op \in \{+, |\} \quad \text{where } (P', \beta'') = \mathsf{RN}(P, \beta),\ (Q', \beta') = \mathsf{RN}(Q, \beta'')$$

$$\mathsf{RN}(A, \beta) \stackrel{\text{def}}{=} (A, \beta), \mathbf{if}\ A \stackrel{\Delta}{=} P$$

Fig. 6. Definition of renaming function RN

B Detailed proofs

B.1 Proof of Lemma 2

Part (i): By induction on the structure of Φ.
Case Φ of
tt: trivial, since $AMC(N, \beta), s \models \mathrm{tt}$ and $\chi(s) \in B(p_1, \ldots, p_n)$
$\aleph_j$:

$AMC(N, \beta), s \models \aleph_j$

$\equiv$ {Def of χ}

$\chi(s) = z_1 \ldots z_n$, with $z_j = p_j$

$\equiv$ {Def of $R(\aleph_j)$}

$\chi(s) \in R(\aleph_j)$

$\neg\Phi$:

$\mathsf{PCF}(\neg\Phi) \ \wedge \ AMC(N,\beta), s \models \neg\Phi$

$\equiv$ {Def of PCF, Def of $\models$}

$\mathsf{PCF}(\Phi) \ \wedge \ \neg(AMC(N,\beta), s \models \Phi)$

$\equiv$ {Def of χ, I.H. }

$\mathsf{PCF}(\Phi) \ \wedge \ \chi(s) \in B(p_1,\ldots,p_n) \ \wedge \ \chi(s) \notin R(\Phi)$

$\equiv$ {Def of $R(\mathrm{tt})$, Properties of Boolean Algebra}

$\mathsf{PCF}(\Phi) \ \wedge \ \chi(s) \in R(\mathrm{tt}) \setminus R(\Phi)$

$\equiv$ {Def of PCF, Def of R}

$\mathsf{PCF}(\neg\Phi) \ \wedge \ \chi(s) \in R(\neg\Phi)$

$\Phi \vee \Phi'$:

$\mathsf{PCF}(\Phi \vee \Phi') \ \wedge \ AMC(N,\beta), s \models \Phi \vee \Phi'$

$\equiv$ {Def of PCF, Def of $\models$}

$\mathsf{PCF}(\Phi) \ \wedge \ \mathsf{PCF}(\Phi') \ \wedge \ (AMC(N,\beta), s \models \Phi \ \vee \ AMC(N,\beta), s \models \Phi')$

$\equiv$ {I.H. }

$\mathsf{PCF}(\Phi) \ \wedge \ \mathsf{PCF}(\Phi') \ \wedge \ (\chi(s) \in R(\Phi) \ \vee \ \chi(s) \in R(\Phi'))$

$\equiv$ {Def of PCF, Set Theory, Def of R}

$\mathsf{PCF}(\Phi \vee \Phi') \ \wedge \ \chi(s) \in R(\Phi \vee \Phi')$

$\mathcal{S}_{\bowtie p}(\Phi), \mathcal{P}_{\bowtie p}(\Phi \,_\xi\mathbf{U}^{<t}\, \Phi'), \mathcal{P}_{\bowtie p}(\Phi \,_\xi\mathbf{U}^{<t}_{\xi'}\, \Phi')$: trivial in all these cases PCF evaluates to false.

Part (ii): trivially follows from Part (i) and from the fact that if $\mathsf{PCF}(\Phi) = \mathrm{ff}$ then, by definition of R, $R(\Phi) = R(\mathrm{tt}) = B(p_1,\ldots,p_n)$ and, by definition of χ, $\chi(s) \in B(p_1,\ldots,p_n)$ Q.E.D.

B.2 Proof of Theorem 1

By induction on the structure of Φ. For simplicity we show the proof only for the untimed cases.
Case Φ of
tt: trivial.
$\aleph_j$:

$AMC(N,\beta), s \models_{MoSL-} \aleph_j$

$\equiv$ {Lemma 2(i)}

$\chi(s) \in R(\aleph_j)$

$\equiv$ {Def of A}

$\forall \gamma \in Act.\ \mathsf{cod}(\gamma, \chi(s)) \in A(\mathrm{tt}, \aleph_j)$

$\equiv$ {Def of $FAMC(N,\beta)$}

$\forall \sigma_F \in \mathsf{Paths}_F(s).\ \alpha(\sigma_F, 0) \in A(\mathrm{tt}, \aleph_j)$

$\equiv$ {Def of $\models$, $\mathbf{X}$, $FAMC(N,\beta)$; See remark on fairness in Sect. 4}

$FAMC(N,\beta), s \models_{aCSL} \mathcal{P}_{\geqslant 1}(\mathbf{X}_{A(\mathrm{tt},\aleph_j)}\ \mathrm{tt})$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\aleph_j)$

$\neg\Phi$:

$AMC(N,\beta), s \models_{MoSL^-} \neg\Phi$

$\equiv$ {Def of $\models$}

$\neg(AMC(N,\beta), s \models \Phi)$

$\equiv$ {I.H. }

$\neg(FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\Phi))$

$\equiv$ {Def of $\models$}

$FAMC(N,\beta), s \models_{aCSL} \neg\mathsf{T}(\Phi)$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\neg\Phi)$

$\Phi \vee \Phi'$:

$AMC(N,\beta), s \models_{MoSL^-} \Phi \vee \Phi'$

$\equiv$ {Def of $\models$}

$(AMC(N,\beta), s \models_{MoSL^-} \Phi) \vee (AMC(N,\beta), s \models_{MoSL^-} \Phi')$

$\equiv$ {I.H. }

$(FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\Phi)) \vee (FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\Phi'))$

$\equiv$ {Def of $\models$}

$FAMC(N,\beta), s \models_{aCSL} (\mathsf{T}(\Phi) \vee \mathsf{T}(\Phi'))$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\Phi \vee \Phi')$

$\mathcal{S}_{\bowtie p}(\Phi)$:

$AMC(N,\beta), s \models_{MoSL^-} \mathcal{S}_{\bowtie p}(\Phi)$

$\equiv$ {Def of $\models$}

$\lim_{t\to\infty} \Pr\{\sigma \in Paths(s) \mid AMC(N,\beta), \sigma(t) \models_{MoSL^-} \Phi\} \bowtie p$

$\equiv$ {I.H. }

$\lim_{t\to\infty} \Pr\{\sigma_F \in Paths_F(s) \mid FAMC(N,\beta), \sigma_F(t) \models_{aCSL} \mathsf{T}(\Phi)\} \bowtie p$

$\equiv$ {Def of $\models$}

$FAMC(N,\beta), s \models_{aCSL} \mathcal{S}_{\bowtie p}(\mathsf{T}(\Phi))$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\mathcal{S}_{\bowtie p}(\Phi))$

$\mathcal{P}_{\bowtie p}(\Phi\, _\xi\mathbf{U}\, \Phi')$:

$AMC(N,\beta), s \models_{MoSL^-} \mathcal{P}_{\bowtie p}(\Phi\, _\xi\mathbf{U}\, \Phi')$

$\equiv$ {Def of $\models$}

$\Pr\{\sigma \in Paths(s) \mid \exists k \geqslant 0.\ AMC(N,\beta), \sigma[k] \models_{MoSL^-} \Phi' \ \wedge$

$\forall 0 \leqslant i < k.\ (AMC(N,\beta), \sigma[i] \models_{MoSL^-} \Phi \wedge \alpha(\sigma,i) \models_{MoSL^-} \xi)\} \bowtie p$

$\Rightarrow$ {I.H., Lemma 3, Def of $FAMC(N,\beta)$}

$\Leftarrow$ {I.H., Lemma 1}

$\Pr\{\sigma_F \in Paths_F(s) \mid \exists k \geqslant 0.\ FAMC(N,\beta), \sigma_F[k] \models_{aCSL} \mathsf{T}(\Phi') \ \wedge$

$\forall 0 \leqslant i < k.\ (FAMC(N,\beta), \sigma_F[i] \models_{aCSL} \mathsf{T}(\Phi) \wedge \alpha(\sigma_F, i) \in A(\xi, \Phi)\} \bowtie p$

$\equiv$ {Def of $\models$}

$FAMC(N,\beta), s \models_{aCSL} \mathcal{P}_{\bowtie p}(\mathsf{T}(\Phi)\ {}_{A(\xi,\Phi)}\mathbf{U}\ \mathsf{T}(\Phi'))$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathcal{P}_{\bowtie p}(\mathsf{T}(\Phi\ {}_{\xi}\mathbf{U}\ \Phi'))$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\mathcal{P}_{\bowtie p}(\Phi\ {}_{\xi}\mathbf{U}\ \Phi))$

$\mathcal{P}_{\bowtie p}(\Phi\ {}_{\xi}\mathbf{U}_{\xi'}\ \Phi')$:
$AMC(N,\beta), s \models_{MoSL^-} \mathcal{P}_{\bowtie p}(\Phi\ {}_{\xi}\mathbf{U}_{\xi'}\ \Phi')$

$\equiv$ {Def of $\models$}

$\Pr\{\sigma \in Paths(s) \mid \exists k \geqslant 0.\ AMC(N,\beta), \sigma[k] \models_{MoSL^-} \Phi' \ \wedge$

$AMC(N,\beta), \sigma[k-1] \models_{MoSL^-} \Phi \ \wedge \alpha(\sigma, k-1) \models_{MoSL^-} \xi' \wedge$
$\forall 0 \leqslant i < k-1.\ (AMC(N,\beta), \sigma[i] \models_{MoSL^-} \Phi \wedge \alpha(\sigma, i) \models_{MoSL^-} \xi)\} \bowtie p$

$\Rightarrow$ {I.H., Lemma 3, Def of $FAMC(N,\beta)$}

$\Leftarrow$ {I.H., Lemma 1}

$\Pr\{\sigma_F \in Paths_F(s) \mid \exists k \geqslant 0.\ FAMC(N,\beta), \sigma_F[k] \models_{aCSL} \mathsf{T}(\Phi') \ \wedge$

$FAMC(N,\beta), \sigma_F[k-1] \models_{aCSL} \mathsf{T}(\Phi) \ \wedge \alpha(\sigma_F, k-1) \in A(\xi', \Phi) \wedge$
$\forall 0 \leqslant i < k-1.\ (FAMC(N,\beta), \sigma_F[i] \models_{aCSL} \mathsf{T}(\Phi) \wedge \alpha(\sigma_F, i) \in A(\xi, \Phi)\} \bowtie p$

$\equiv$ {Def of $\models$}

$FAMC(N,\beta), s \models_{aCSL} \mathcal{P}_{\bowtie p}(\mathsf{T}(\Phi)\ {}_{A(\xi,\Phi)}\mathbf{U}_{A(\xi',\Phi)}\ \mathsf{T}(\Phi'))$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathcal{P}_{\bowtie p}(\mathsf{T}(\Phi\ {}_{\xi}\mathbf{U}_{\xi'}\ \Phi'))$

$\equiv$ {Def of T}

$FAMC(N,\beta), s \models_{aCSL} \mathsf{T}(\mathcal{P}_{\bowtie p}(\Phi\ {}_{\xi}\mathbf{U}_{\xi'}\ \Phi))$ Q.E.D.